WHAT IS CLAIMED IS:

1. A method for communicating with a device, comprising:

executing a kernel module in a memory;

executing at least one kernel thread in the memory to handle calls to device driver functions for the kernel module; and

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context.

2. The method of claim 1, wherein the kernel module spawns at least one kernel thread to execute the calls to the device driver functions for the kernel module.

3. The method of claim 1, further comprising:

accessing, with one kernel thread, device information from the device; and

buffering the accessed device information.

4. The method of claim 3, wherein a kernel module function requests device information, further comprising:

in response to the request for the device information, accessing the buffered device information.

5. The method of claim 1, wherein the kernel thread accesses buffered device information periodically and independently of kernel module requests for the device information.

6. The method of claim 1, further comprising:

buffering a parameter list;

setting device parameters in the buffered parameter list to values provided by kernel module functions.

7. The method of claim 6, further comprising:

setting a flag indicating that the kernel thread needs to set parameters at the device to device parameter values set in the parameter list.

8. The method of claim 6, further comprising:

spawning a kernel thread to set device parameters to parameter values buffered in the parameter list.

9. The method of claim 7, wherein the kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

10. The method of claim 7, wherein the kernel thread processes the parameter list by further performing:

applying a lock on information in the parameter list including the located buffered parameter values;

after applying the lock, copying the parameter values from the parameter list to a temporary buffer, wherein the device parameters are set to the parameter values from the parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the parameter list to the temporary buffer.

11. The method of claim 10, further comprising:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

12. The method of claim 11, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

13. The method of claim 10, further comprising:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

14. The method of claim 1, further comprising:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

15. A system, comprising:

a network device;

a memory;

a processor executing code to perform:

(i) execute a network device driver in memory to control the network device;

(ii) execute a kernel module in the memory;

(iii) execute at least one kernel thread in the memory to handle calls to device driver functions for the kernel module; and

(iv) execute, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context.

16. The system of claim 15, wherein the kernel module spawns at least one kernel thread to execute the called device driver functions.

17. The system of claim 15, wherein the processor executes code to further perform:

access, with one kernel thread, device information from the device; and

buffer the accessed device information.

18. The system of claim 17, wherein a kernel module function requests device information, wherein the processor executes the code to further perform:

in response to the request for the device information, accessing the buffered device information.

19. The system of claim 15, wherein the kernel thread accesses device information periodically and independently of kernel module requests for device information.

20. The system of claim 15, wherein the processor executes the code to further perform:

buffering a parameter list; and

setting device parameters in the buffered parameter list to values provided by kernel module functions.

21. The system of claim 20, wherein the processor executes the code to further perform:

setting a flag indicating that the kernel thread needs to set parameters at the device to device parameter values set in the parameter list.

22. The system of claim 21, wherein the kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

23. The system of claim 21, wherein the executed kernel thread processes the parameter list by further performing:

applying a lock on information in the parameter list including the located buffered parameter values;

after applying the lock, copying the parameter values from the parameter list to a temporary buffer, wherein the device parameters are set to the parameter values from the parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the parameter list to the temporary buffer.

24. The system of claim 23, wherein the processor executes the code to further perform:

disabling higher priority context before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

25. The system of claim 23, wherein the processor executes the code to further perform:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

26. The system of claim 15, wherein the processor executes the code to further perform:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

27. An article of manufacture for communicating with a device, wherein the article of manufacture causes operations to be performed, the operations comprising:

executing a kernel module;

executing at least one kernel thread to handle calls to device driver functions for the kernel module; and

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context.

28. The article of manufacture of claim 27, wherein the kernel module spawns at least one kernel thread to execute the called device driver functions.

29. The article of manufacture of claim 27, wherein the operations further comprise:

accessing, with one kernel thread, device information from the device; and

buffering the accessed device information.

30. The article of manufacture of claim 29, wherein a kernel module function requests device information, wherein the operations further comprise:

in response to a request for the device information, accessing the buffered device information.

31. The article of manufacture of claim 27, wherein the kernel thread accesses buffered device information periodically and independently of kernel module requests for device information.

32. The article of manufacture of claim 27, wherein the operations further comprise:

buffering a parameter list;

setting device parameters in the buffered parameter list to values provided by kernel module functions.

33. The article of manufacture of claim 32, wherein the operations further comprise:

setting a flag indicating that the kernel thread needs to set parameters at the device to device parameter values set in the parameter list.

34. The article of manufacture of claim 27, wherein the kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

35. The article of manufacture of claim 34, wherein the kernel thread processes the parameter list by further performing:

applying a lock on information in the parameter list including the located buffered parameter values;

after applying the lock, copying the parameter values from the parameter list to a temporary buffer, wherein the device parameters are set to the parameter values from the parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the parameter list to the temporary buffer.

36. The article of manufacture of claim 35, wherein the operations further comprise:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

37. The article of manufacture of claim 36, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

38. The article of manufacture of claim 35, wherein the operations further comprise:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

39. The article of manufacture of claim 27, wherein the code executes operations to further perform:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.